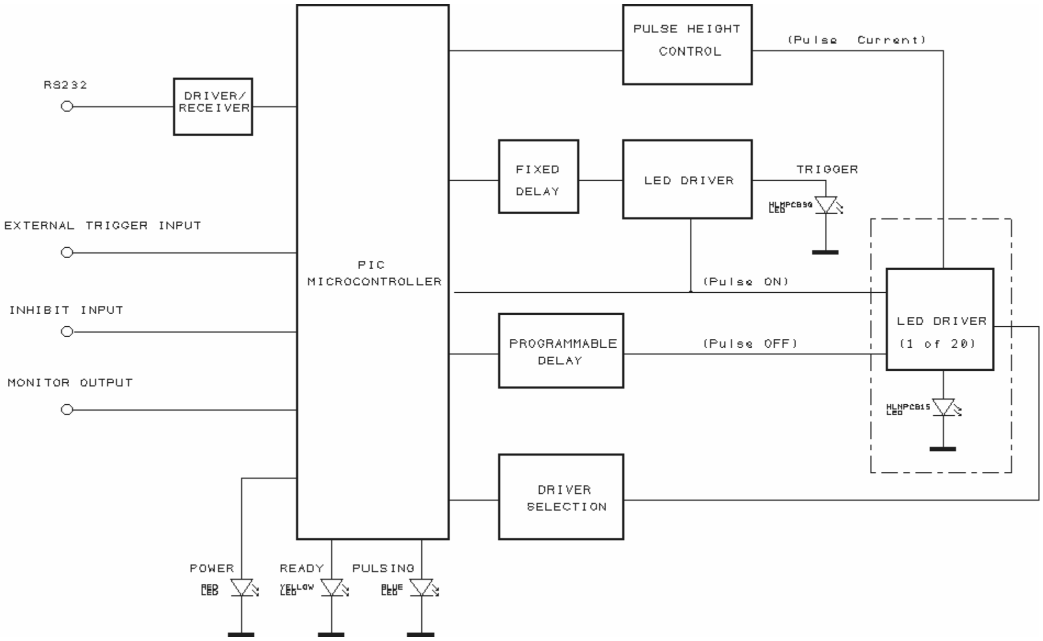
MINOS PULSER BOX ELECTRONICS

1. Overview

Each pulser box can drive up to twenty LEDs. Only one of these LEDs can be pulsed at any one time. Pulse widths can be set from 15 to 36nS, and pulse height selected via a 12 bit digital to analogue converter. The pulsing can be in a continuous or preset count mode at a pre-selected repetition rate. A trigger LED is provided that pulses in synchronisation[1] with a selected LED. Front panel indicators show the status of the system, and BNC connectors are provided for the following: monitoring the pulsing signal, external triggering and inhibit signal input.

Control of all functions is via an RS232 serial connection to the onboard microcontroller, which in sets the requested height, width and repetition rate for a selected LED.



A pulser box has one control card, which feeds two LED driver cards. Each driver card has ten outputs and is completely interchangeable since the control card determines address decoding. The control card has two output edge connector sockets, a driver card plugged into the upper socket drives LED's 1 to 10 and a driver card plugged into the lower socket drives LED's 11 to 20. The system can be run with only one driver card connected if required.

Note 1. Far detector main pulse delayed 245nS with respect to trigger pulse.
Near detector main pulse has a selectable delay of zero to 290nS.

2. Control Card

The main component is U1, is a PIC 16F877 microcontroller running at a clock speed of 4MHz. Serial communication is via U2. The PCB was designed to accommodate a MAX489 IC in this position for RS485, but the standard was subsequently revised to RS232 requiring a modification to the PCB (Note 7.2.1). Inputs MCLR, RB6 and RB7 are reserved for onboard programming and are not normally used. An address switch is connected to ports RC0 to RC4, originally designed to identify each Pulser box, this function is now accomplished through the Ethernet to RS232 converter, and for normal operation the address switch should be set to 111110.

Varying the voltage applied to a 22R series resistor sets LED current and thus the pulse height. This voltage is set by the programmable voltage source formed by the serial digital to analogue converter, U3, dual op-amp U1a/b and Darlington transistor Q1. The controlled voltage can be set from 4 to 12 volts, using the full range of the 10-bit digital to analogue converter.

A LED pulse is accomplished as follows: Output RE2 is set high, turning on the current to the selected LED via edge connector pins 6 and 27, simultaneously with the onboard trigger LED current switch formed by Q6 and Q3. After a delay of 2uS, giving time for the current to stabilise, RE1 is set high to start the pulse via U4c, U5a out to edge connector pins 7 and 26. Provision is made for an inhibit signal at one input of U5a, a low on this input immediately halts pulsing. Input RE0 enables the controller to sense this event so that it can resume any preset sequence when this line is released without losing count of pulses in a sequence. The programmable delay line U9 determines the on time of the pulse, after the selected delay plus a short fixed delay through U2 the current to the selected LED via edge connector pins 8 and 25 is turned off. In order to ensure that the trigger LED is coincident with a main LED pulse it to is driven by the same output lines as above but with the following minor differences. The pulse period delay is provided by U2, this is a fixed time of 30nS. The LED current is set by a fixed 15v into 56R. An additional TTL pulse monitor output is provided via U5 b, c and d.

Provision has been made for an external trigger pulse. This input requires a high to low transition and will use preloaded pulse height, width and LED selections previously sent to the microcontroller. The monostables U8 and U16 switch on the current and initiate the pulse in place of RE1 and RE2 via the OR gates U4c and U4d. Note that the front panel ‘Pulsing’ indicator is only active during software-instigated runs and does not indicate externally triggered events.

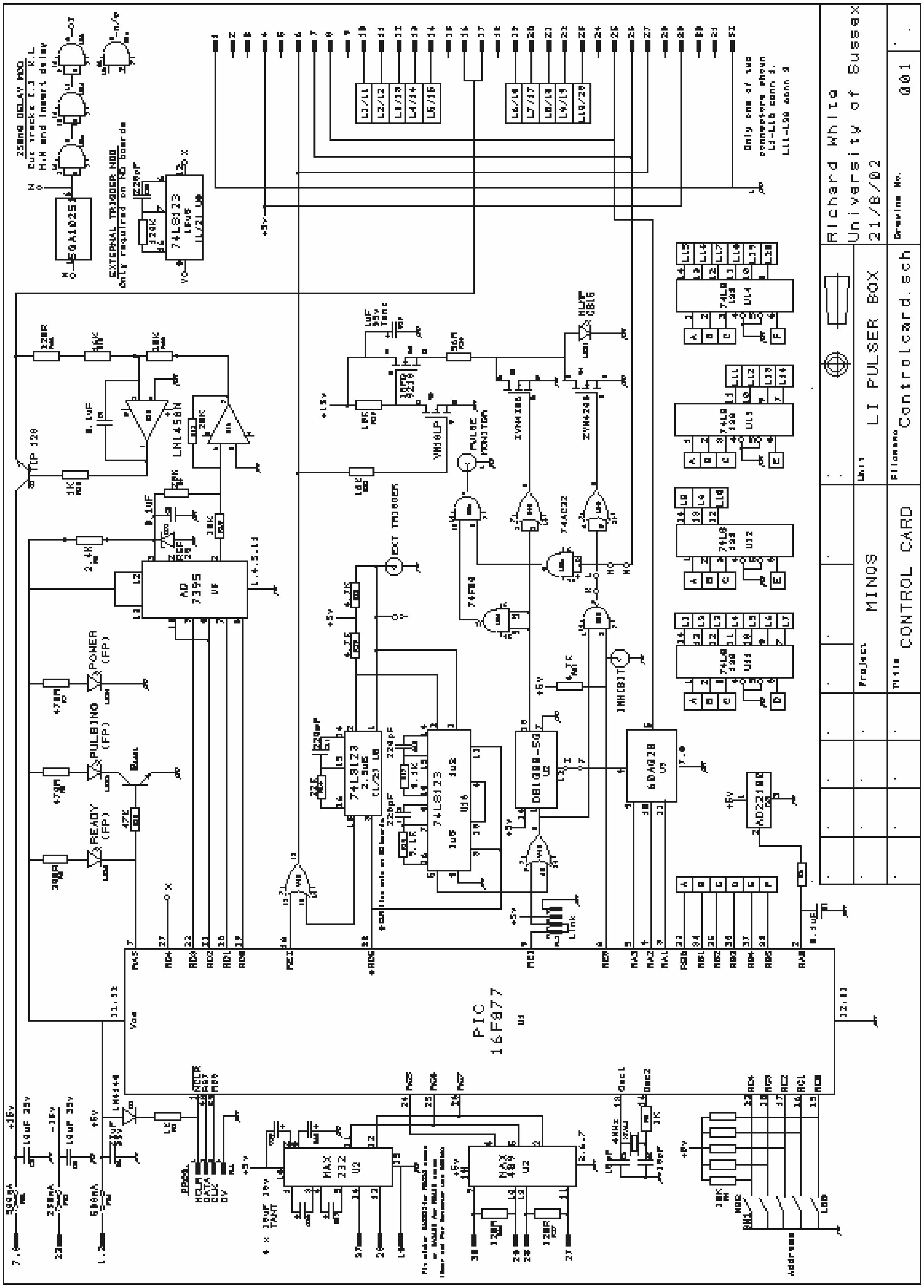
Unique selection decoding for each LED is achieved via the 3 to 8 line decoders, U11 to U14, with outputs L1 to L10 routed to the upper connector, and outputs L11 to L20 routed to the lower connector.

Q3 is a voltage output temperature sensor that can be used to monitor conditions inside the pulser box if required.

The Far detectors Pulser Boxes have a requirement for a 250nS delay between LED pulses and the trigger pulse. This has required the addition of a small board and the cutting of two tracks (Note 7.2.2).

The Near Detector boards have been modified to allow the number of accepted external trigger pulses to be preset. (Note 7.2.4). This also required a firmware update to P12. The trigger pulse delay has also been made more flexible by the addition of a switch selectable delay line (Note 7.2.3). This has an overall range of 0 to 290nS. This switch must always have one of 1 to 5 ON (0 to 40nS in 10nS steps) and one of 6 to 11 ON (0 to 250nS in 50nS steps). Note that switch 12 is not used in this application.

Note: there is a four-way pin connector below U4, labelled GATE, this must have a shorting link fitted over the first two pins on the left hand side when viewing the board from the component side.


Richard White
University of Sussex
21/8/02
Drawing No.
001
Unit
LI PULSER BOX
Filename
Controlcard.sch
Project
MINOS
Title
CONTROL CARD
PIC 16F877
MAX 232
MAX 489
AD 7395
LM1458N
74LS123
DS1000-50
6DAG10
AD22100
TIP 120
VN10LP
ZVN4206
ZVN4210
74AC32
74F00
EXT TRIGGER
PULSE MONITOR
INHIBIT
POWER (FP)
PULSING (FP)
READY (FP)
250nS DELAY MOD
EXTERNAL TRIGGER MOD
Only required on ND boards
Only one of two connectors shown
Address
Link

3. LED Driver Boards

Fig. 2 shows one of ten identical LED driver stages from an LED driver board and its associate drive waveforms.

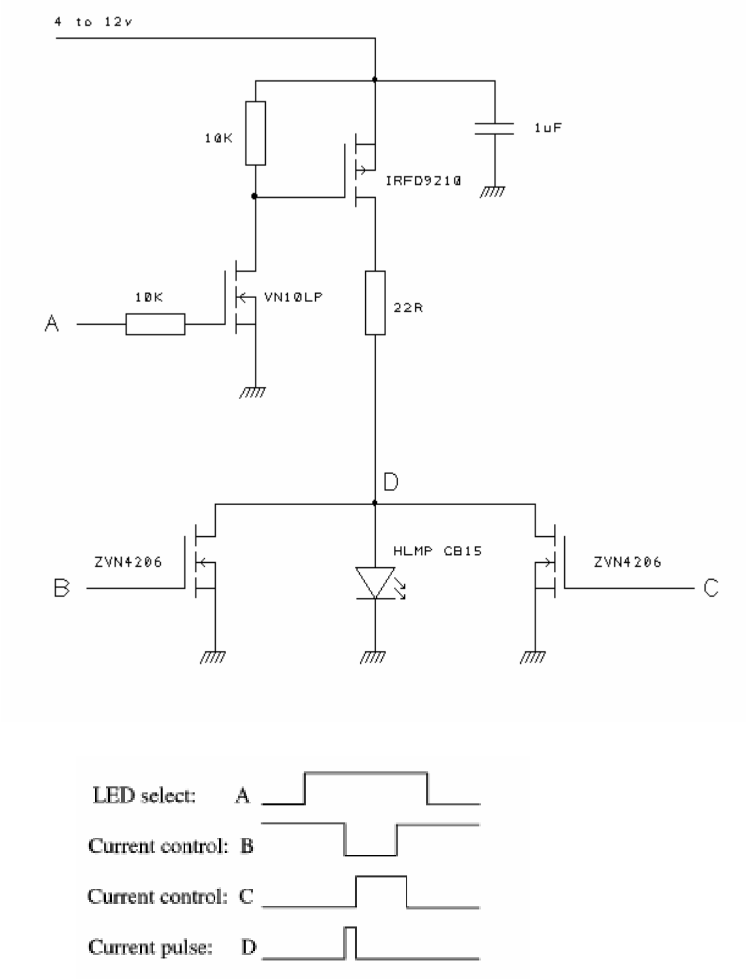
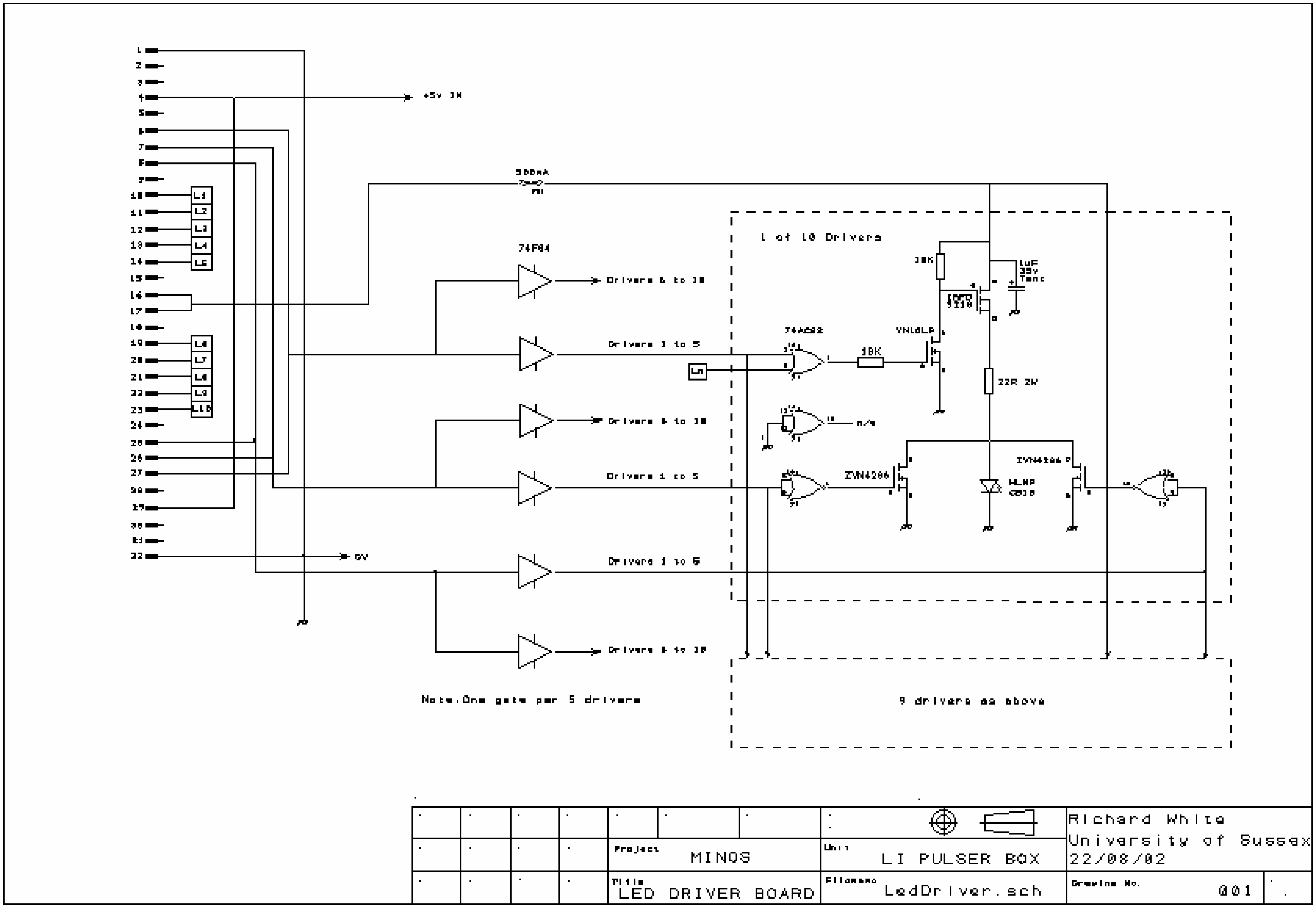


Fig 2.

The applied voltage, which determines the pulse height, is controlled by a 10-bit DAC over the range 4 to 12v. When the signal at A goes high, the driver stage is turned on—voltage is applied to the 22R resistor, the resulting current is shunted to ground via the left-hand FET. This occurs typically a microsecond before the actual pulse, to give the current time to stabilise. When the signal at B goes low, current flows through the LED, which emits light. A short time later, signal C goes high, and the current is shunted to ground via the right-hand FET; the LED is then turned off. The delay between B turning off and C turning on determines the pulse width; a programmable delay line on the control board, gives a pulse width of 15–36 ns in 3 ns steps. The current continues to be shunted to ground until the signal at A goes low again some 1.5uS after the end of the light pulse, removing the current. Other components on the board are 74AC02s used as MOSFET drivers for each stage and two 74F04s employed as buffers for the signal lines, one for each group of 5 drivers per board.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
L1 L2 L3 L4 L5
L6 L7 L8 L9 L10
+5v IN
0V
500mA
F1
74F04
Drivers 6 to 10
Drivers 1 to 5
Drivers 6 to 10
Drivers 1 to 5
Drivers 1 to 5
Drivers 6 to 10
Ln
Note,One gate per 5 drivers
1 of 10 Drivers
10K
1uF
35v
Tant
IRFD
9110
74AC32
VN10LP
10K
22R 2W
n/c
ZVN4206
HLMP
CB10
ZVN4206 P
9 drivers as above
Richard White
University of Sussex
22/08/02
Project
MINOS
Unit
LI PULSER BOX
Title
LED DRIVER BOARD
Filename
LedDriver.sch
Drawing No.
001

4.Control card tests

Using a test lead as defined in appendix 1, make connections to a suitable power supply. The maximum current consumption is around 200 mA for each of the positive supplies and less than 10 mA for the negative supply. A three wire serial connection should also be made between the board under test and a PC or laptops serial port1. When the supplies are switched on, the Power and Ready LEDs should illuminate.

A simple Visual Basic programme, Test.exe, is available to exercise all the functions of the board. First ensure that the board address switch is set to 1 (11110), and then send the command **STARTUP**, the **REPLY** box should display ‘OK’. This simple test verifies the communication link to the board, and subsequent commands should be received with no problems.

Test the onboard trigger LED by first entering the pulse to pulse delay by settting the **DEL L** (delay low) slider to 10, then entering this value by selecting **DEL L**. The **DEL H** setting defaults to 1 and can be left untouched. Now select **Continuous**, the trigger LED should illuminate. Now select **Stop** to extinguish the led. *Note: the pulser should always be stopped before new commands are entered.* Increasing **DEL L** to 255 should visibly dim the LED.

Pressing the Inhibit switch whilst the pulser is running will stop any LED pulsing, this can be confirmed by observing the trigger LED which will be extinguished until the switch is released. Observe the monitor socket with an oscilloscope, a 30nS negative going TTL signal coincident with each light pulse should be seen. The control card incorporates a temperature sensor which can be read at any time the when pulsing is not active. To complete functional checks of pulse height and width settings, an LED driver card must be attached, see next section.

5. LED Driver card tests

Note: Turn off all power to the control card before plugging in a driver card.

As previously stated control cards have two output edge connector sockets, a driver card plugged into the upper socket drives LED's 1 to 10 and a driver card plugged into the lower socket drives LED's 11 to 20. For initial tests it will be found best to use the lower card position, then the assembly will stand conveniently on the workbench.

The driver card can be loaded with any number of LEDs from one to ten, and no harm will occur if none are fitted. Looking into the sockets the lowest numbered socket will be on the left for either driver card.

Now reapply power and check that communication is re-established by sending the **STARTUP** command once again. Before an LED can be pulsed the following parameters must be set-up via the control programme:

1. **Box number** - defaults to 1, no need to enter, provided the board address switch is set to 1.
2. **LED number** - Remember the lower card carries LEDs 11 to 20. Remember to press button to load.
3. **Pulse Width** – Select from 0 to 7, 0=min 7=max.
4. Height is selected by two bytes, **Hhigh** and **Hlow** the height is displayed as a decimal value in the Height box, after entering these numbers they must be loaded into the converter by pressing **DAC**. It is possible to verify this action by checking the voltage at Fuse1, located at the centre of the driver card. This has a range of 4 to 12v over the full range of height settings.
5. Set the pulse-to-pulse delay as previously. **DEL H**=1, default, **DEL L**=10, this is the minimum delay that can be safely applied and will give enough light from the LED to be readily visible.
6. **NUM H** and **NUM L** are used to enter the number of pulses in a sequence. Note that NUM H is a multiplier for NUM L and with a short pulse-to-pulse delay these will have to be quite high to enable the LED pulse to be seen for more than a few seconds.
7. At this point all the parameters have been loaded, now all that remains is to select **Continuous**, and the selected LED should illuminate. Always terminate this action with **STOP** for a safe exit. Always stop the pulsing before changing any parameters, including reading the **Temperature**.
8. **Sequence** will pulse the LED using the selected parameters for the preset number of pulses entered in step 6. A sequence run can be terminated at any time if required using **STOP**.
9. Test all LED drivers by selecting the appropriate **LED number** and repeating steps 3 to 9 as required.

6. Control card versions

The control card PCB was originally designed for RS485 communications, with the option of a relatively easy modification (Note 7.2.1) to RS232 if required. The majority of pulser boxes now have RS232 as the common standard, the only boxes still using the RS485 standard are the Cal Det (CERN), and Near Detector 40 LED box, in the latter case this is only for internal communications, the outside world still sees the box interface as RS232.

An addition to all control cards was the incorporation of a delay in the output of the main pulse with respect to the trigger pulse. This modification required the addition of a small board to the lower right hand edge of the control card, and this exists in two versions, a fixed 245nS delay. (Note 7.2.2) and a variable 0 to 290nS delay with 10nS steps. (Note 7.2.3)

A specific requirement for the near detector boards was the ability to define a preset number of trigger pulses. To achieve this goal a modification to the external trigger circuit was made (Note 7.2.4).

The last major modification proposed for the control cards was the extension of the pulse width settings available to give more light if required. By changing the existing 3-wire parallel programmable delay line (60A-028) to a serial programmed delay line (DS 10215-50) it was possible to extend the selection of widths available. In order to keep compatibility with the original boards, widths 0 to 7 are the same, and control software written for the original version of the board will work ok. The delay steps are in increments of 3nS up to 50nS, then in 25nS steps up to a maximum of 143nS. (Note 7.2.6).

During the assembly of the boards the DS100-50 delay line became obsolete. A suitable replacement was obtained (DS1100-50) but unfortunately this was only available in an 8-pin package, so necessitates modification to the control card. Only a few of these were fitted, but if future repairs to boards require replacement of this part then follow the instructions given (Note 7.2.5).

Some of the modifications mentioned above require new versions of the firmware. The actual version of firmware is clearly marked on the 40-pin microcontroller chip (e.g. P10.) The following table lists the modifications and firmware in use at the various sites.

Detector	**Firmware**	**Trigger delay**	**External trigger**	**Extended pulse width**
Far	P10	245nS	Standard	No
Near	P12	0- 290nS	Software preset	No
CalDet	P14	0- 290nS	Software preset	Yes

For all versions the onboard address switch must always be set to 1 (off), the only exception to this rule is the lower most board in the 40 LED Near detector box, which is set to 2 (off).

7. PCB Modifications

7.1: Modification to LED Driver card

This is required on all driver boards to correct a design error.

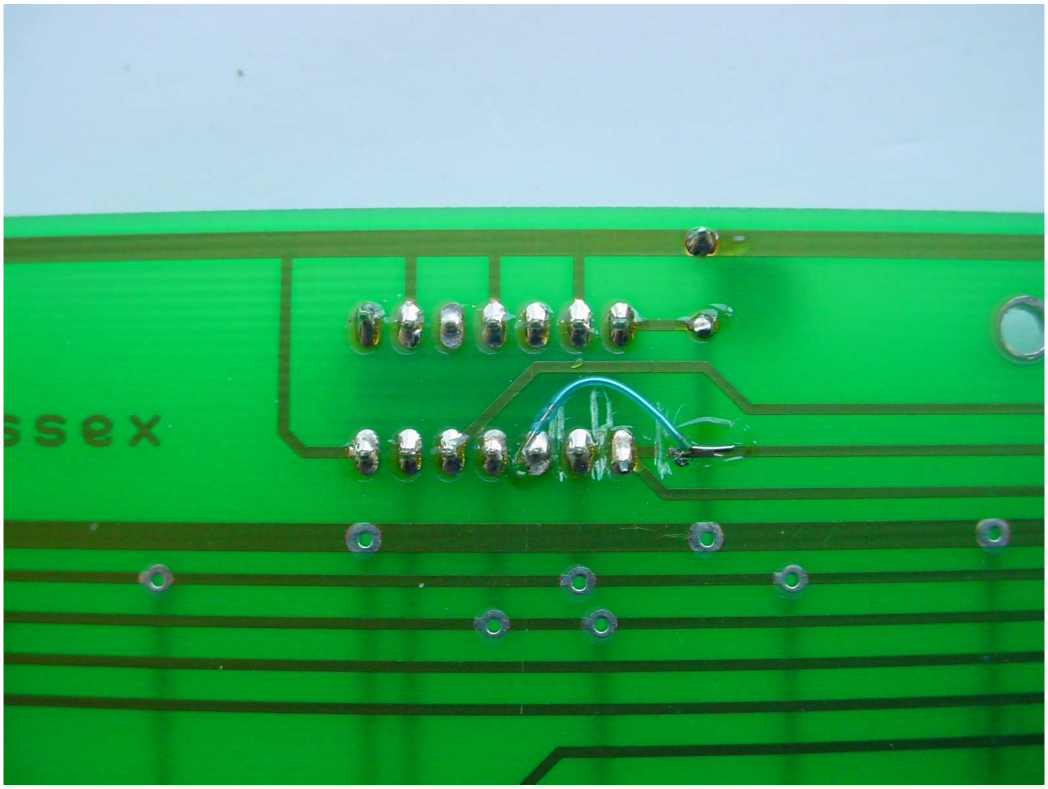


Fig.4

Cut the three tracks as shown in photo:

From edge connector pin 26 to pin 14 on U12
Between pin 12 and pin 13 on U12
Between pin 13 and pin 14 on U12

Add a wire link from edge connector track (pin 26) to U12 pin 12.

7.2: Modifications to LED Control board

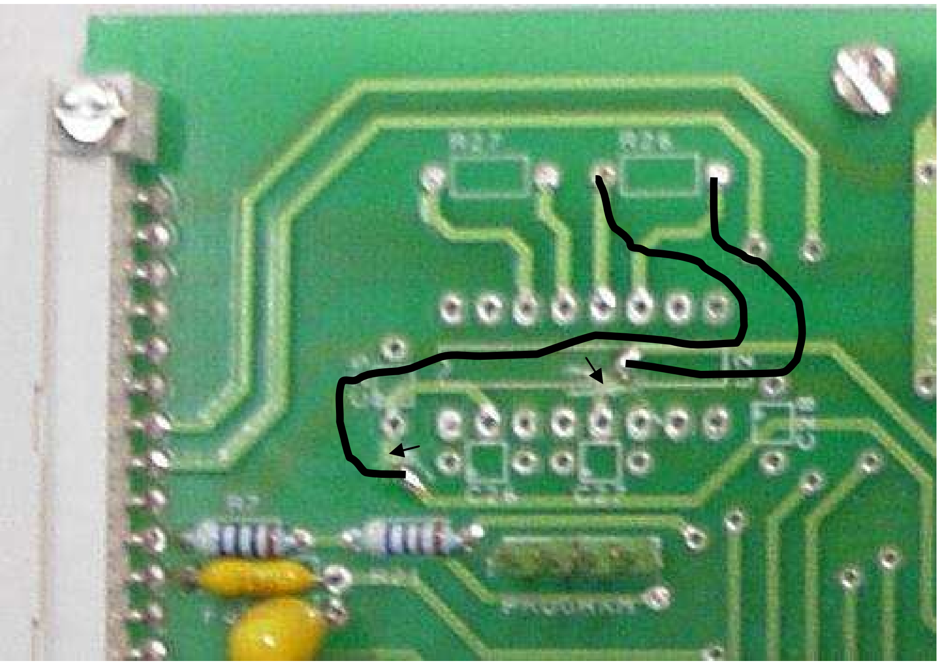


Fig.5

7.2.1: Conversion to RS232

The standard LED Control board was primarily designed for an RS485 communications link, so normally a MAX 489 IC would be in the U2 position and R26 & R27, both 120R, would be fitted.

In the final system a RS232 communication was implemented via an Ethernet adaptor for each Pulser Box. These are installed at both the Near and Far Detectors.

To convert the board to RS 232 the following modifications are required:

1. Remove IC. MAX 489 (U2) if fitted.

2. Remove R26 & R27 (120R) if fitted.

3. Cut the tracks on the top side of the board as indicated by the arrows
   U1, P25 to U2, P5
   U1, P26 to U2, P2

4. Now link track from U1, P25 to R26 right hand end (connects to U2, P11)
   And link track from U1, P26 to R26 left hand end (connects to U2, P12)

5. Fit 10uF 16v Tant. Caps as C25, C26, C27 and C28. Follow marked polarity.

Now do the modifications to the underside of the board before fitting the MAX 232.

Conversion to RS 232 – Underside mods.

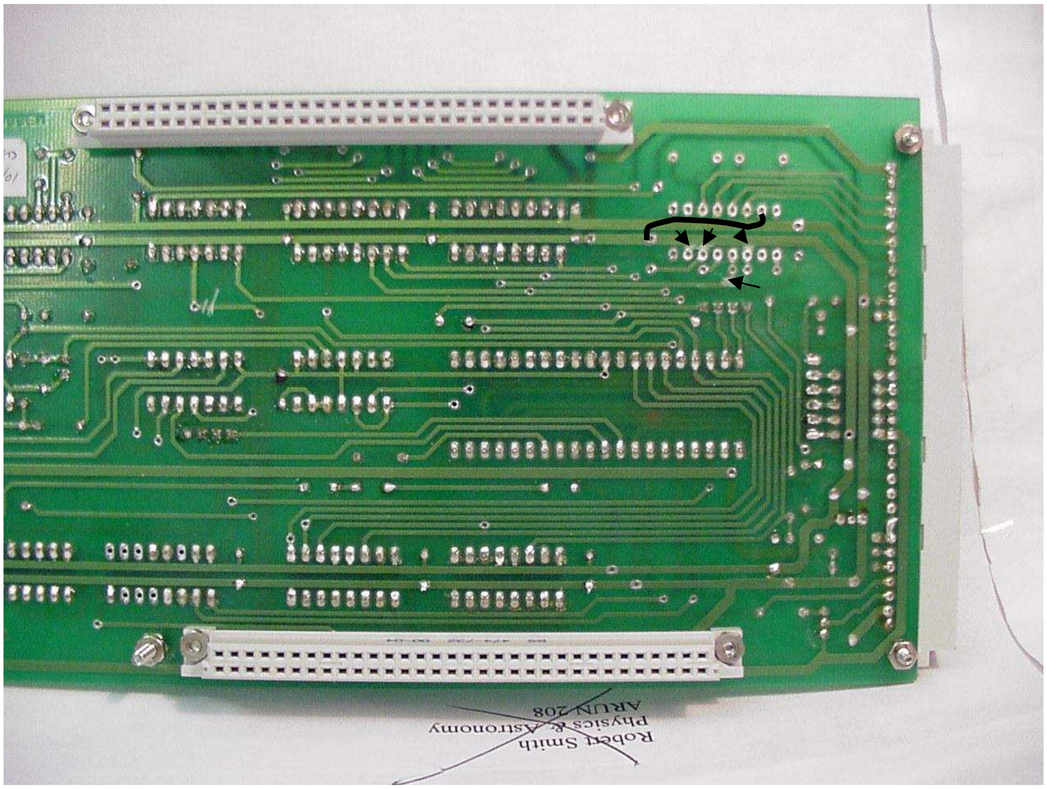


Fig. 6.

6. Cut three connections to the OV. Track U2, pins 3, 6 and 7 as arrowed.

7. Also cut the track to U2, pin 4, below the C27 connection.

8. Now fit the MAX 232 IC. In the U2 position pin 1 to pin 1, this leaves two vacant holes (8 & 9) since this IC has only 14 pins.

9. Finally fit a link from the OV track to U11 pin 15 (or pin 13 on the MAX 232)

7.2.2: Adding the 250nS Delay board to the Control card

Cut track from U5 pin 4, on the right hand side of the plated through hole.

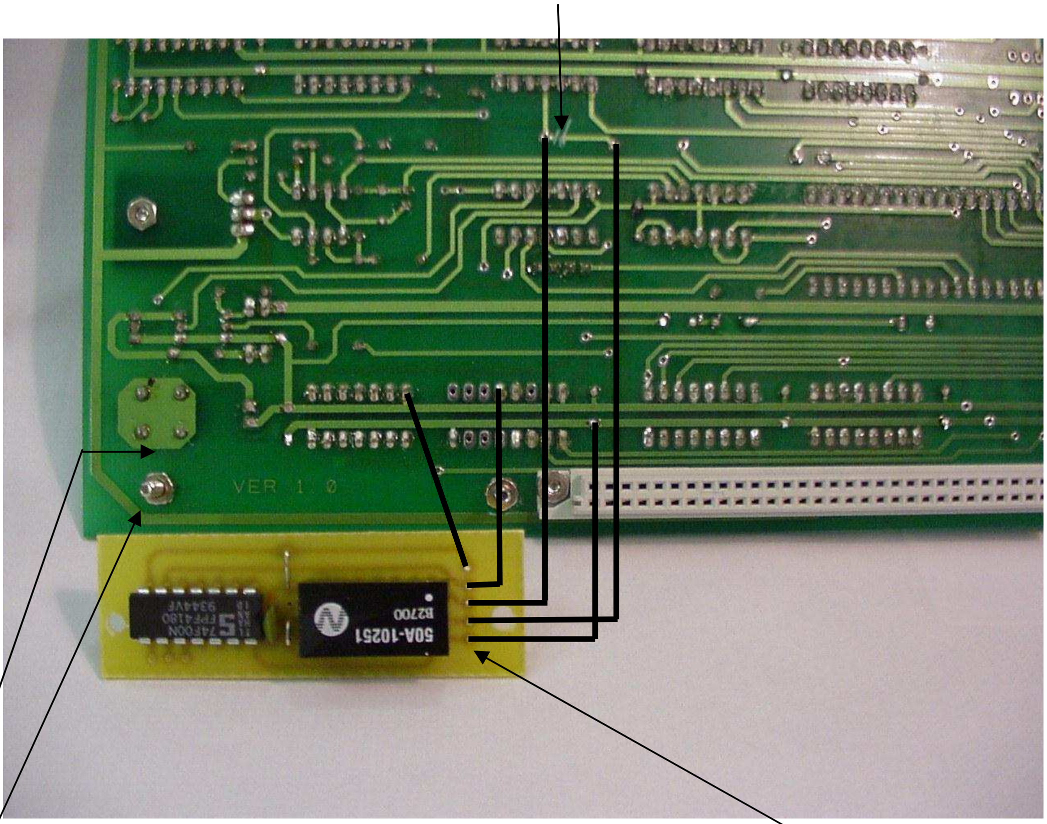


Fig.7

Pin 1

M2.5 X 12 with 2 nuts under each end of the board as spacers.

0.5 inch from edge of board.

Link:

1. Delay board Pin1 to +5v.
2. Delay board Pin 2 to 32 way skt. Pin 7.
3. Delay board Pin 3 to 74F00 (U5) Pin 4.
4. Delay board Pin 4 to 60A028 (U9) Pin 4.
5. Delay board Pin 5 to 0v.

Now cut track on topside of board – see next page.

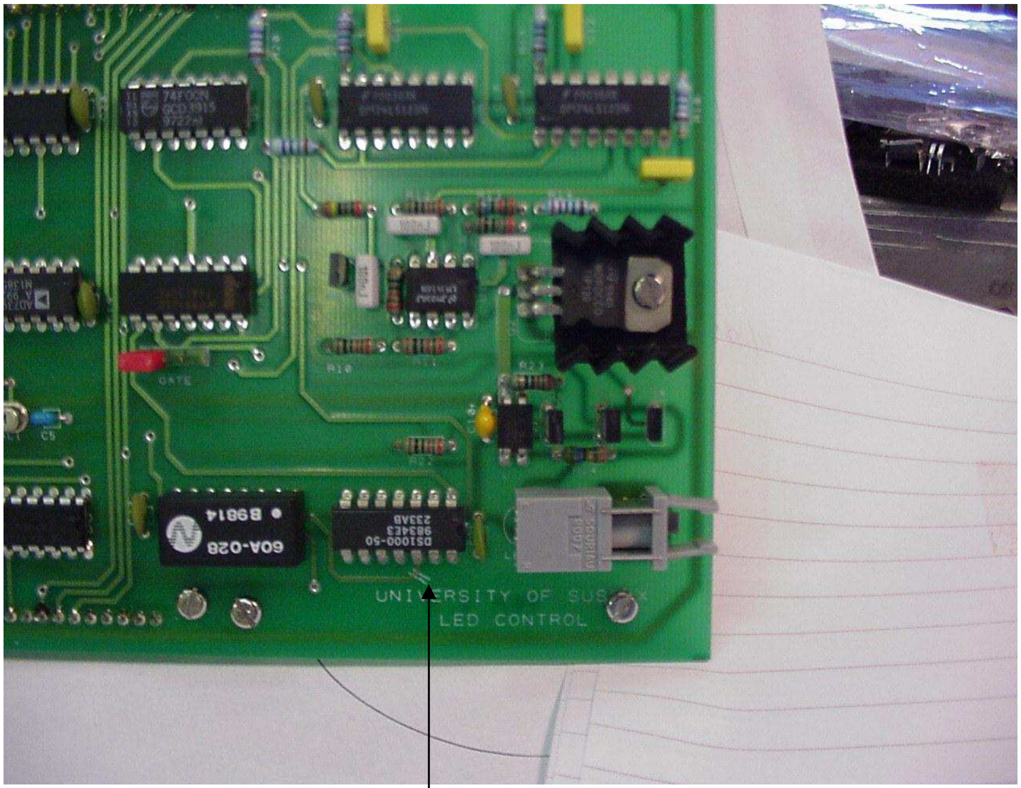


Fig. 8.

Cut track to DS1000-50 at Pin 12

7.2.3: Adding the variable Delay board to the Control card

The variable delay board is assembled on the single sided PCB (Appendix PCB-4)
Note that the 12-way switch is actually mounted on the trackside of the board.

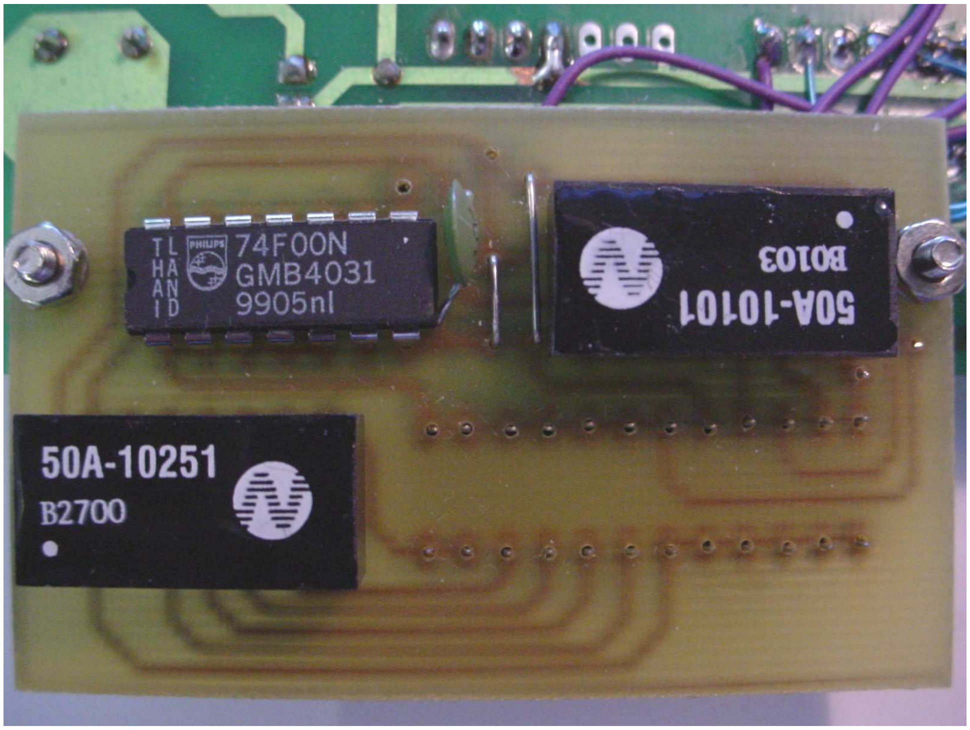
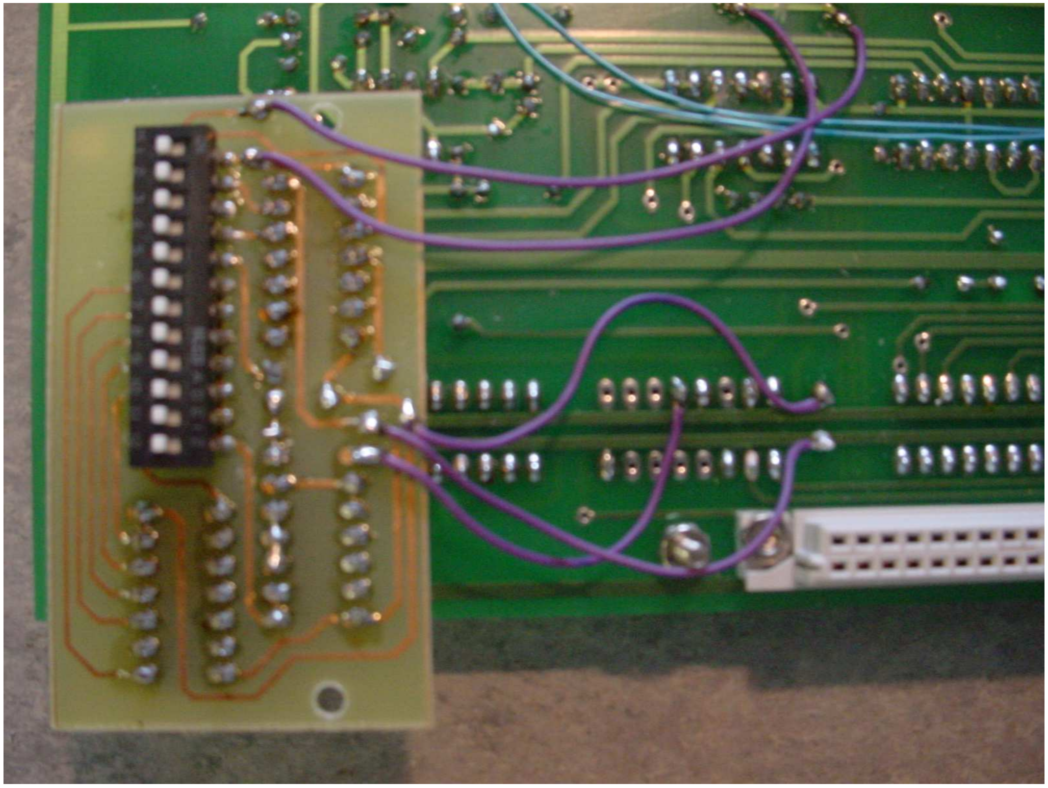


Fig.9.



Track cut here

Fig.10.

Once the wiring has been completed the board is turned over and mounted on two screws in the same manner as the previously shown fixed delay board. The switch is then accessible from the topside as shown overleaf.

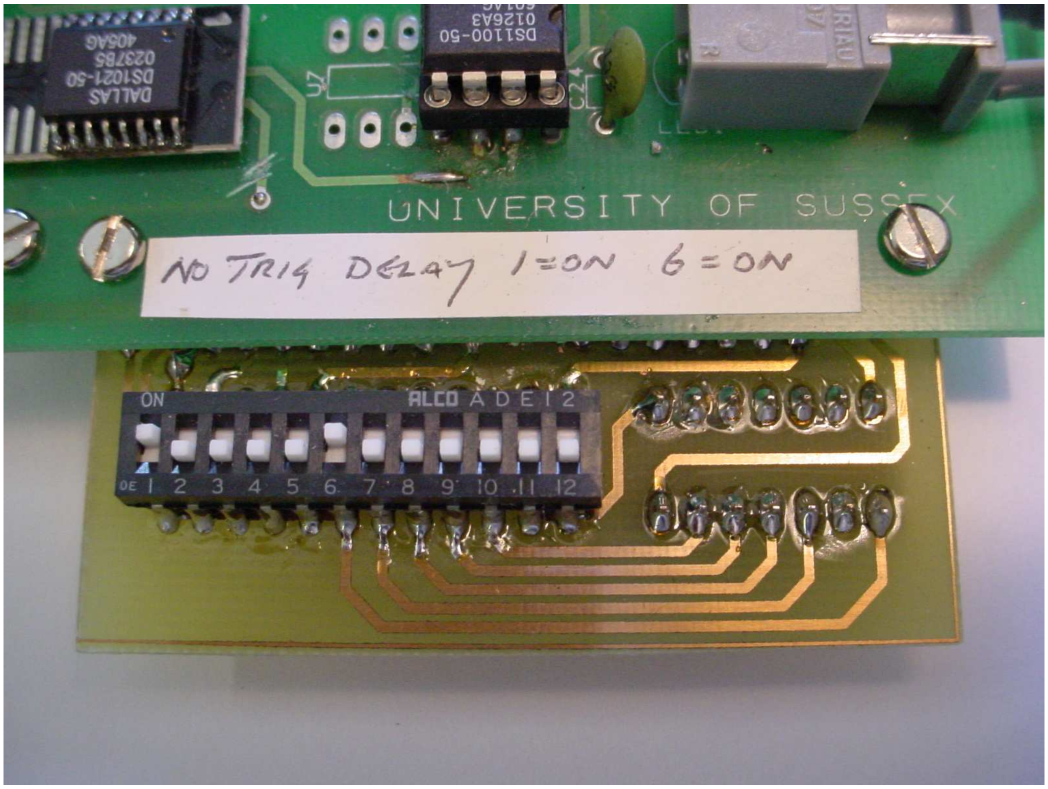


Fig. 11.

The delay switch is used to combine the delays from two delay line IC's. Switches 1 to 5 give a delay of 0 to 40nS in 10nS steps and switches 6 to 11 give a delay of 0 to 250nS in 50nS steps. Ensure that only one switch of each group is on at a time. Fig.11. illustrates no delay selected (sw1=on, sw2=on). Note that switch 12 is connected in parallel with switch 11 and is not used. Although primarily designed for use at the Near detector, a control card fitted with this switched delay can be used at the far detector by matching the delay time. This is achieved by selecting sw1=on and sw10=on.

7.2.4: External Trigger modification for Near Detector boards.

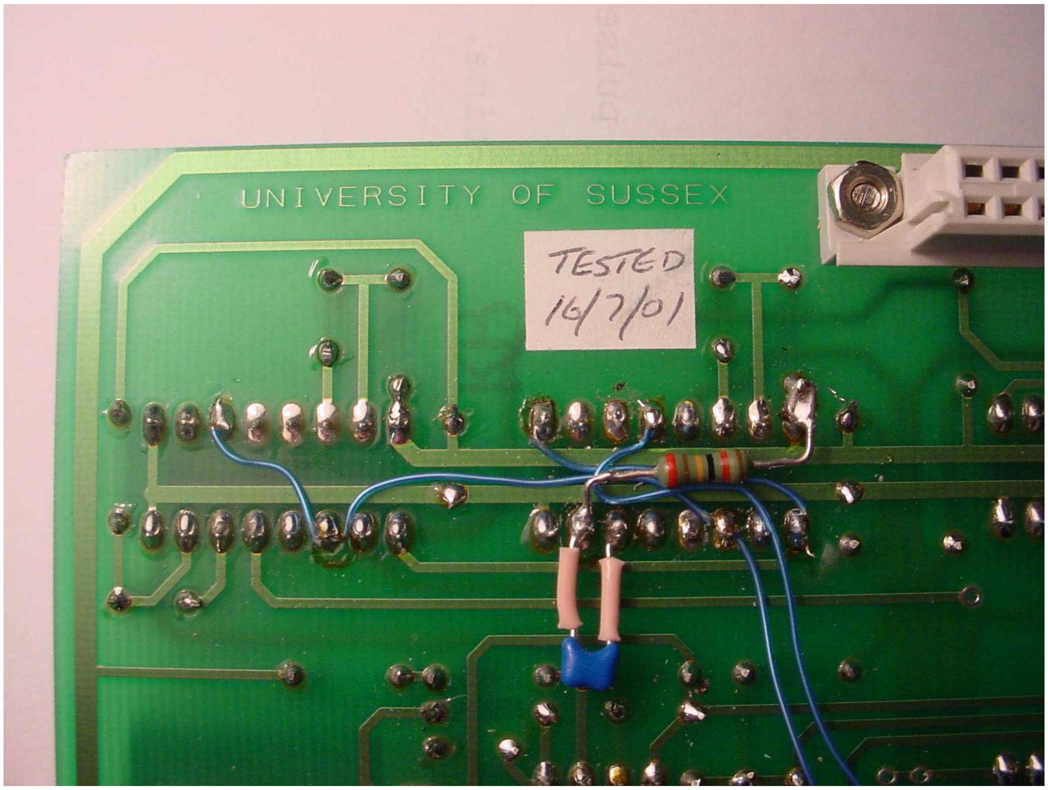


Fig. 12.

Note: No tracks to cut.

1. Link U9 Pin 3 to U9 Pin 11, to U8 Pin 3, to U1 Pin 28.
2. Link U8 Pin 1 to U8 Pin 9.
3. Link U8 Pin 12 to U1 Pin 27.
4. Add 220pF between U8 Pins 6 and 7.
5. Add 120K between U8 Pins 7 and 16.

Full view of the External trigger mod and Far detector trigger delay

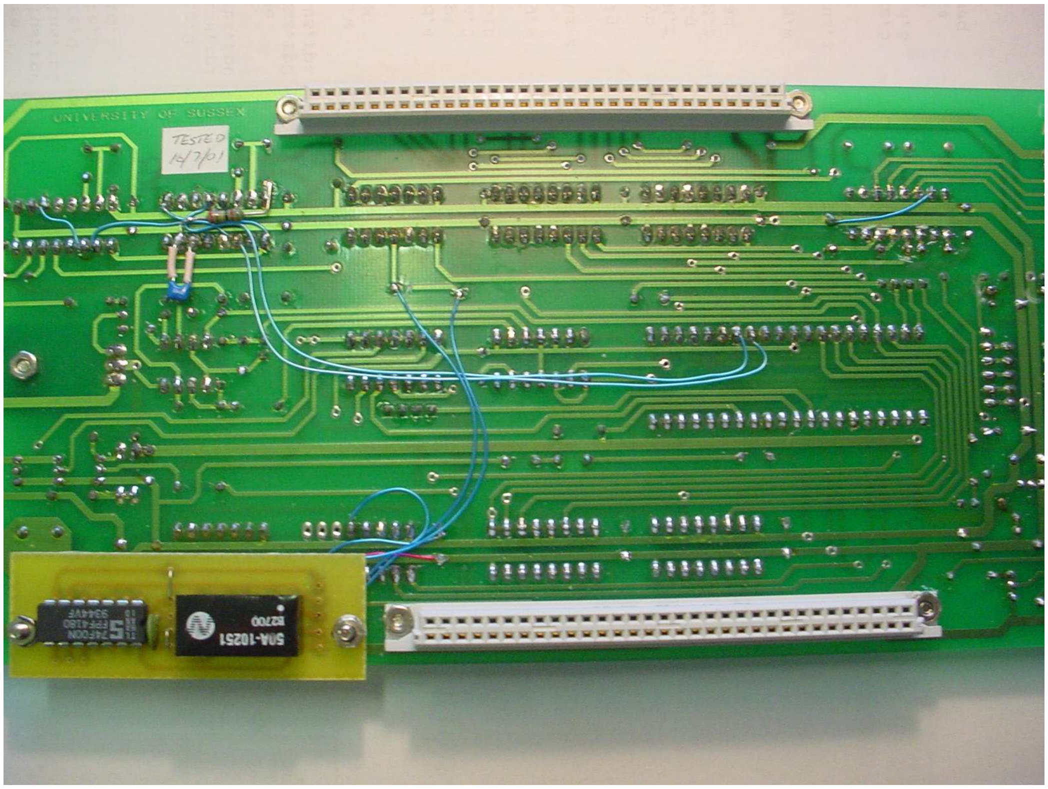


Fig. 13.

7.2.5: Fitting the DS1100-50 (replaces obsolete delay line DS1000-50)

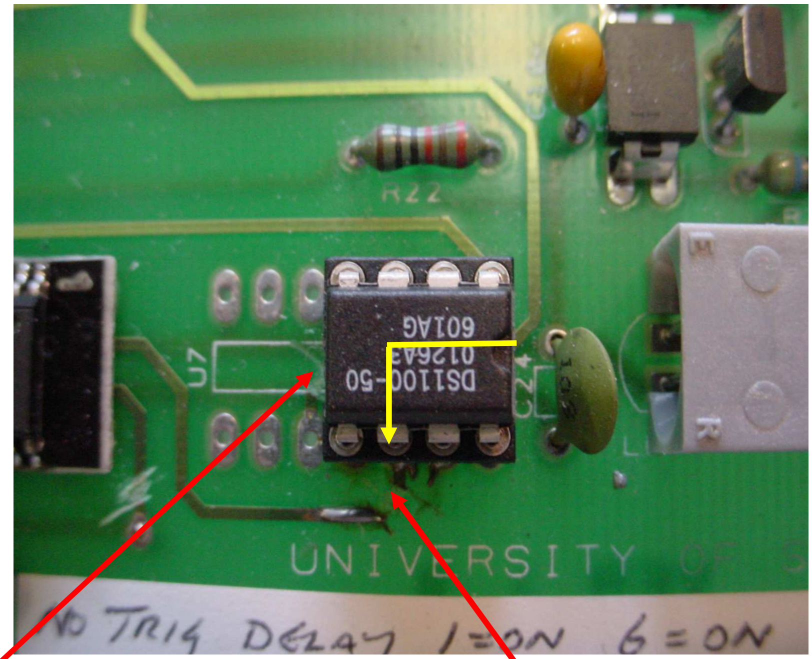


Fig. 14. Cut track here.

Cut track here (pin 10 on original 14 pin chip) and link back to pin 6 of the replacement DS1100-50, as shown by the yellow arrow. This is the track which routes under the IC and exits at the top edge of the picture.

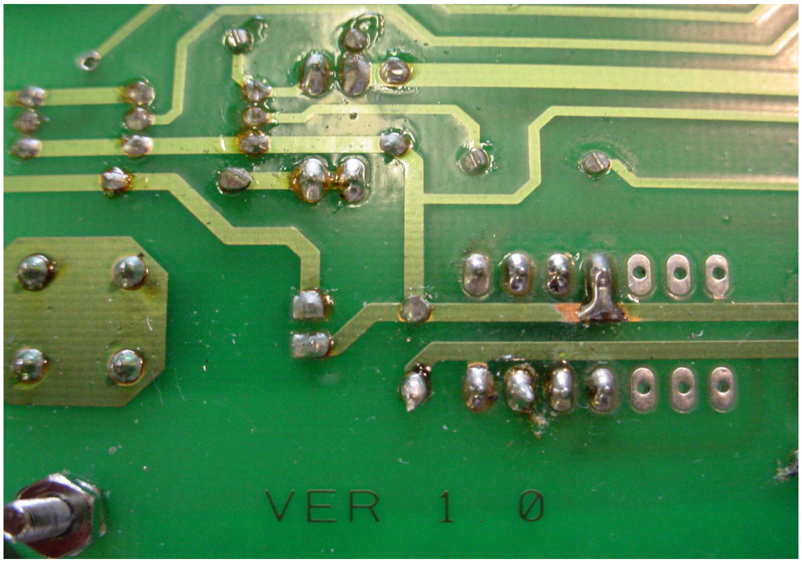


Fig. 15.

On the under side of the board make a link from the OV track to pin 4 of the DS1100-50, after removing lacquer coat.

7.2.6: Extended pulse width range

If this facility is required, first remove the 60A-028 delay line.
Now carefully mount the DS1021-50 on to a DIL adaptor. Positioning can be aided by the use of a small drop of SM adhesive under the centre of the package. Use a fine soldering iron, any bridging can be successfully removed with solder wick.

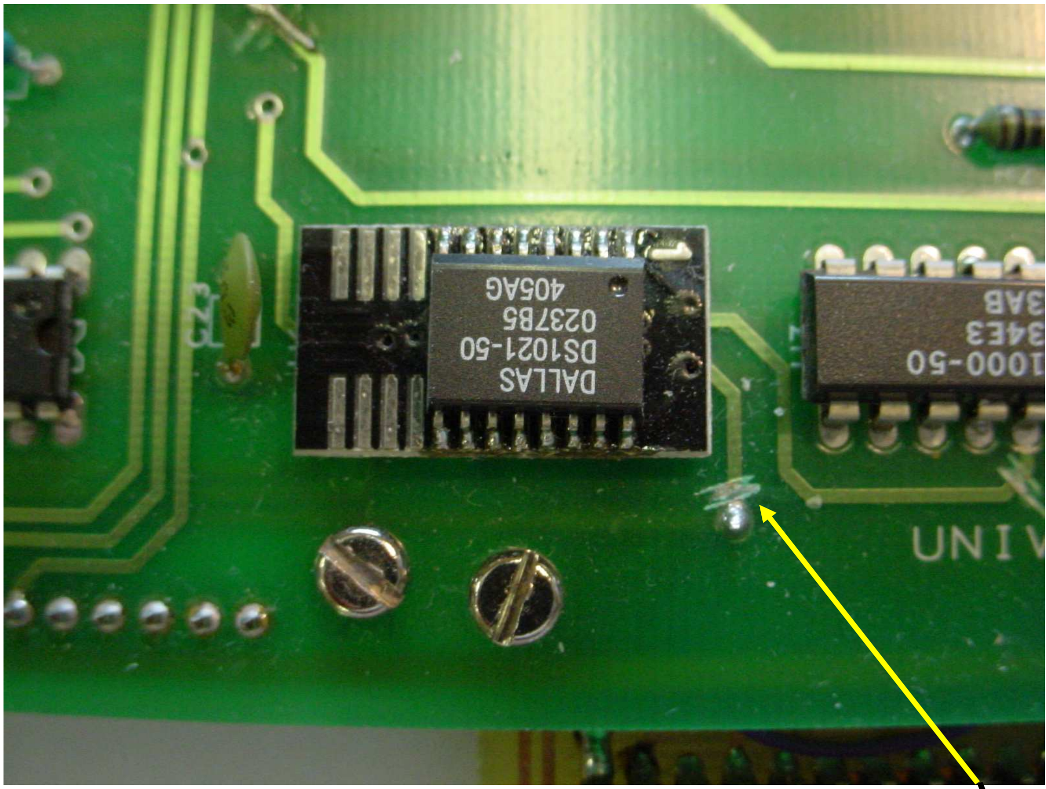


Fig. 16.

Cut track here, then solder in adaptor.

Note: Track is also cut here for the trigger delay mod.

The violet wires belong to a previously fitted trigger delay board. The indicated wire shown in Fig. 10 as routed to pin 4, now must be connected to pin1. All new wiring for this modification is in blue.

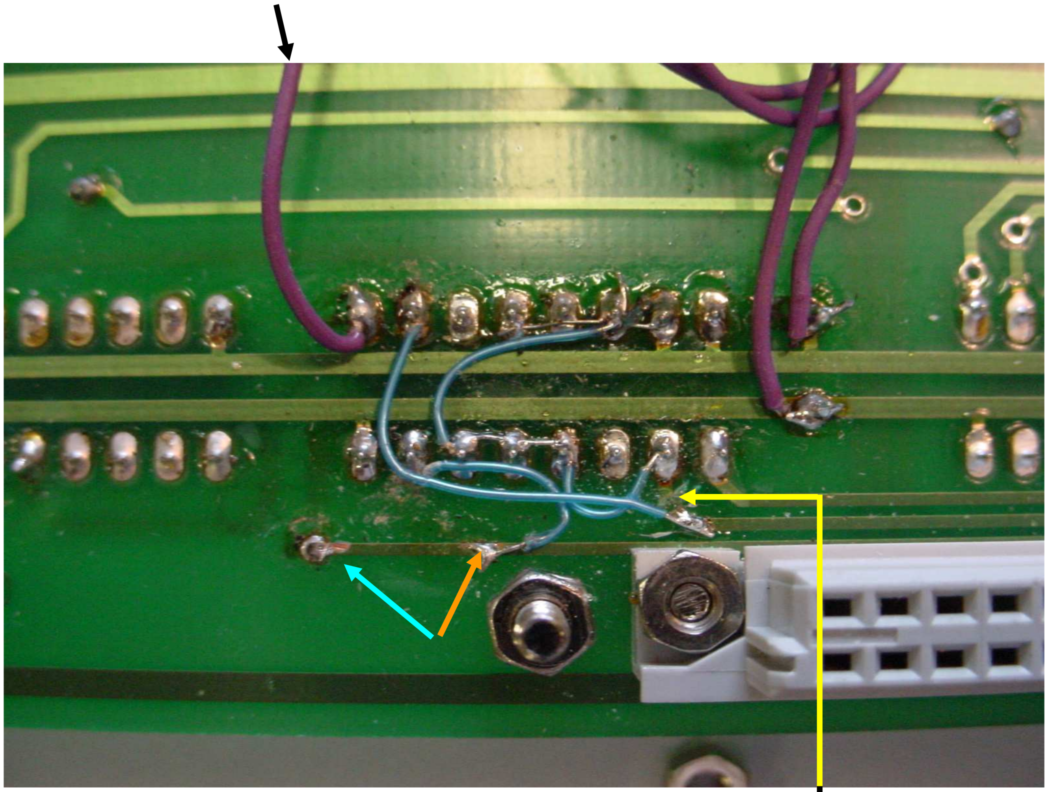


Fig.16.

Cut track here.

Underside wiring: Connect pins 4,5,6,10,12,13 and 14 to OV (Pin 7).
Leave pin 3 unconnected.
Rewire from cut track below yellow arrow to pin 2.
Run wire from lower track (orange arrow) to pin 15. When modifying boards in the future it would be easier to connect to the solder pad at the end of this track as indicated by the blue arrow.

Appendix 1: Control card test lead

Construct a test lead by wiring a 32 way DIN 4162 single row socket as follows:

1. +5v
2.
3. Green LED cathode (Pulsing indicator)
4. Yellow LED cathode (Ready indicator)
5. Green LED anode (Pulsing indicator)
6. Yellow LED anode (Ready indicator)
7. +15v
8.
9.
10.
11. Reset switch pin1 (Normally open)
12.
13. Inhibit switch pin1 (Normally open)
14. Reset switch pin2 (Normally open)
15. Inhibit switch pin2 (Normally open)
16. External trigger coax socket ground
17. Monitor coax socket ground
18. Red LED cathode (Power indicator)
19.
20.
21.
22. –15v
23. Red LED anode (Power indicator)
24.
25. Monitor coax socket centre pin
26. External trigger coax socket centre pin
27. Serial port1, pin 2
28. Serial port1, pin 3
29.
30.
31. Serial port1, pin 5
32. OV

Appendix 2: Firmware

Code for the PIC16F877 was written in assembler, using the free Microchip MPLAB integrated development system and compiled into HEX code for programming using the same.

The original firmware labelled P10 is now superseded by P12, which can be used on all control cards except those with the extended pulse width capability, which must use the dedicated firmware P14.

Hex code files for the three versions P10, P12 and P14 are used with a suitable programmer, to programme blank PIC 16F877 microcontrollers. They should in addition be configured to operate with a high-speed crystal oscillator and the watchdog timer should be disabled.

Firmware listing for P12

```
;Addressable Pulser Box with RS485 or RS232 serial link

;Rev1.External pulse counting added
;Rev2.Continuous external pulse mode added

;                  PROGRAMMABLE LED PULSER



;          CODE     FUNCTION        PARAMETERS    REPLY
;        ----------------------------------------------------

;           A         Startup                 none          K

;           B        Pulse height top 2 bits   0-3          K

;           C      Pulse height low byte,    Nh,Nl.         K K [wait 2mS]

;           D        Pulse width               0-7          K

;           E        Select LED                1-20 **      K

;           F      Pulse multiplier,0-255.   Nh,Nl.         K K

;           G      No. of pulses, 0-255.     Nh,Nl.         K K
;        H    Period multiplier,0-255.    Nh,Nl          K K

;        I    pulse period, 0-255.        NH,Nl          K K

;        J    Start sequence                              K

;        K    Start continuous                            K

;        L    Read temperature                            K

;        M    Send temperature high byte                  000000vv

;        N    Send temperature low byte                   vvvvvvvv

;        O    Load D/A with pulse height                  K

;        P    Ext. triggered sequence                     K

;        Q    Continuous Ext. triggering                  K

;       _X    Stop sequence or continuous                 K

;                  * Nh=0-F,   Nl=0-F

;      **ASCII for LEDs 1-7=97-103, 8-14=105-111, 15-20=113-118

;      All commands except stop must be preceded by the box number.
;      Box numbers 0-31 are represented by decimal ASCII codes 96-127.

;-------------------------------------------------------------------------
               list     p=16F877
               include<p16F877.inc>


;-------Register Files----------------------
tmro      equ    0x01    ;0,2 (Page locations)
pcl       equ    0x02    ;0,1,2,3
status    equ    0x03    ;0,1,2,3
fsr       equ    0x04    ;0,1,2,3
porta     equ    0x05    ;0
portb     equ    0x06    ;0,2
portc     equ    0x07    ;0
portd     equ    0x08    ;0
porte     equ    0x09    ;0
pclath    equ    0x0A    ;0,1,2,3
intcon    equ    0x0B    ;0,1,2,3
pir1      equ    0x0C     ;0
rcsta     equ    0x18    ;0
txreg     equ    0x19    ;0
rcreg     equ    0x1A    ;0
```

```
adresh   equ   0x1E   ;
adcon0   equ   0x1F   ;
topt     equ   0x81   ;1,3
trisa    equ   0x85   ;1
trisb    equ   0x86   ;1,3
trisc    equ   0x87   ;1
trisd    equ   0x88   ;1
trise    equ   0x89   ;1
pie1     equ   0x8C   ;1
pie2     equ   0x8D   ;1
pcon     equ   0x8E   ;1
txsta    equ   0x98   ;1
spbrg    equ   0x99   ;1
adresl   equ   0x9E   ;1
adcon1   equ   0x9F   ;1
eedata   equ   0x10C  ;2
eeadr    equ   0x10D  ;2
eedath   equ   0x10E  ;2
eeadrh   equ   0x10F  ;2
eecon1   equ   0x18C  ;3
eecon2   equ   0x18D  ;3

wait     equ   0x20   ;allocate registers to variables
rxser    equ   0x21
intdex   equ   0x22
intdex1  equ   0x23
txdata   equ   0x24
rxdata   equ   0x25
delylp   equ   0x26
lobyte   equ   0x28
hibyte   equ   0x29
pwidth   equ   0x2A
rxtemp   equ   0x27
SLED     equ   0x2B
curnth   equ   0x2C
byte     equ   0x2D
nibble   equ   0x2E
number   equ   0x2F
count    equ   0x30
length   equ   0x31
pspace   equ   0x32
mult1    equ   0x33
mult2    equ   0x34
countx   equ   0x35
rep      equ   0x36
rxnum    equ   0x37
boxnum   equ   0x38
tmpbox   equ   0x3A
sdely    equ   0x3B
HNUM     equ   0x3C
```

```
THNUMequ   0x3D
LNUM   equ   0x3E
TLNUM equ   0x3F
HDEL    equ   0x40
THDEL  equ   0x41
LDEL     equ   0x42
TLDEL   equ   0x43
LEDnum equ  0x44
curntl     equ   0x45
d           equ   0x46
chtmp    equ   0x47
cltmp     equ   0x48  ;Note 7F is the last available register

rp0      equ      0x05
rp1      equ      0x06

w        equ      0    ;Result to go into working register (accumulator)
f          equ      1    ;Result to go into a file register.
c         equ      0    ;Carry flag  (located in STATUS register)
dc       equ      1    ;Digit carry            "
z         equ      2    ;Zero flag              "
pd       equ      3    ;Power Down bit         "
to       equ      4    ;Time-out bit           "

         org      0x04
         org      0x05 ;start of program memory
;-----------------------------------------------------------
;Setup PORTS: 0=output, 1=input

         BCF             status,rp1        ;clear page 2/3
         BSF             status,rp0        ;select page 1
         MOVLW        B'10001110'
         MOVWF        adcon1             ;a/d0 selected, remainder digital
         MOVLW        B'00000001'
         MOVWF        trisa               ;PortA,0 A/D input, remainder outputs
         CLRF           trisb               ;PortB all outputs
         MOVLW        B'10011111'
         MOVWF        trisc               ;PortC,6 inputs, 2 outputs
         MOVLW         B'00010000'
         MOVWF        trisd               ;PortD, 7 outputs,1 input for counter
         MOVLW        B'00000001'
         MOVWF        trise                ;PortE, E0=i/p,E1=o/p
         Movlw          0x01                ;set  prescaler
         movwf topt      ; "    "
         BCF             status,rp0         ;select page 0
         bcf              status,rp1

;Pulsing OFF, drivers OFF

         BCF             porte,2             ;disable LED and driver
```

```
        BCF         porte,1         ;no pulsing
        MOVLW       0
        MOVWF       portd            ;current to zero
        MOVLW       0
        MOVWF       porta            ;drivers off
        MOVLW       0
        MOVWF       portb            ;drivers off
        Movwf       pwidth           ;width to 0
        movwf       curnth          ;current hi-byte to 0

;Deselect all LEDs

        movlw       0x00
        movwf       LEDnum
        movwf       portb            ;All LEDs off


;Set baudrate

        BCF         status,rp1      ;clear page 2/3
        BSF         status,rp0      ;select page 1
        MOVLW       d'25'           ;BRG value for 9600 baud
                                    ;from 4.00Mhz, brgh=1 (from SPRG Arc. prog)
        MOVWF       spbrg           ;put into spbrg reg
        MOVLW       B'00000100'     ;sync=0(bit 4),brgh=1(bit 2)
        MOVWF       txsta           ;put into txsta
        BCF         status,rp0      ;set backto page 0

;Read Address switches

        MOVF        portc,w
        ANDLW       b'00011111'     ;only look at 5 lower bits
        ADDLW       b'01100000'     ;box address as ASCII control code
        MOVWF       boxnum           ;save the pulser address

;-----------------------------------------------------------

         GOTO       Start

;-----------------------------------------------------------

;*******SUB_TXCHARACTER*****

txchar  bsf         PORTC,5         ;Note nxchar only used to setup TX registers
nxchar  MOVLW       0x09            ;setup loop for 9 characters
        MOVWF       intdex
        bcf         STATUS,C
txloop  btfss       STATUS,C
        bcf         PORTC,6
        btfsc       STATUS,C
        bsf         PORTC,6
        call        dely83
        rrf         txdata,1
        decfsz      intdex,1
        goto        txloop
        bsf         PORTC,6
        call        dely83
        bcf         PORTC,5
        return

;*******SUB_RXCHARACTER*****

rxchar  bcf         PORTC,5     ;Put MAX485 into RX mode
sbit    btfsc       PORTC,7
        goto        sbit        ;wait for start bit
        movlw       0x08        ;pick up 8 bits
        movwf       intdex
        call        dely46      ;delay to middle of start bit
rxloop  call        dely83
        nop
        bcf         STATUS,C     ;sample incoming bit
        btfsc       PORTC,7
        bsf         STATUS,C
        rrf         rxdata,1
        decfsz      intdex,1
        goto        rxloop
waitend btfss       PORTC,7
        goto        waitend
        return                  ;on exit character in rxdata

;*******SUB_oK***********

ok      MOVLW       0x4B        ;Load 'K'
        movwf       txdata
        call        txchar      ;and send
        return

;*******SUB_ENDSEQ*******

endseq  MOVLW       0x53        ;Load 'S' to denote end of sequence
        movwf       txdata
        call        txchar      ;and send
        return

;*******SUB_HEIGHTh*********
;sets the top 2 bits on D/A

heighth call        short
        rlf         rxnum,f
        rlf         rxnum,f
        rlf         rxnum,f
        rlf         rxnum,f
        rlf         rxnum,f
        rlf         rxnum,f
        movf        rxnum,w
        movwf       curnth
        call        ok
        return

;*******SUB_HEIGHTl*********
;sets the lower 8 bits on D/A

heightl call        hex         ;Read number 0-255
        movf        byte,w
        movwf       curntl      ;Save the current setting
        return

;*******SUB_NUMBER**********

short   call        rxchar      ;Read number 0-7
        MOVF        rxdata,w
        MOVWF       rxnum       ;copy new data into rxnum
        MOVWF       rxtemp      ;and temp file.
        MOVLW       0x30
        ANDWF       rxtemp,w
        xorlw       0x30
        btfss       status,z    ;Wait for a number
        goto        short
        MOVLW       0x07
        ANDWF       rxnum,w     ;only look at first three bits
        MOVWF       rxnum       ;number in temporary store
        return

;*******SUB_SELECT*********

select  call        rxchar      ;Read the LED selection 0-20 (0=LEDs OFF)
        movf        rxdata,w    ;LEDs are represented by ASCII code groups.
        movwf       SLED        ;LEDs 0-7 = 96-103
        btfsc       SLED,4      ;LEDs 8-14 = 105-111
        goto        HiNUM       ;LEDs 15-20 = 113-118
        btfsc       SLED,3      ;Groups are detected by looking at bits 3 & 4
        goto        MidNUM      ;of the ASCII code, this separates them into
        movlw       0x07        ;high,mid or low for the relative decoders.
        andwf       SLED,w
        movwf       LEDnum
        movlw       b'00001000'
        iorwf       LEDnum,f
        goto        selLED
HiNUM movlw         0x07
        andwf       SLED,w
        movwf       LEDnum
        movlw       b'00100000'
        iorwf       LEDnum,f
        goto        selLED
MidNUM movlw        0x07
        andwf       SLED,w
        movwf       LEDnum
        movlw       b'00010000'
        iorwf       LEDnum,f
        selLED      nop             ;Selected LED in LEDnum
        call        ok
        return

;*******SUB_PULSE**********

pulse   bsf         porte,1          ;Output pulse ,approx 2us. Bit set used to
        NOP                         ;avoid conflict with serial data on RA2 & RA3
        bcf         porte,1         ;End pulse
        call        ok
        return

;*******SUB_DELAY46***

dely46  MOVLW       0x0E
delex  MOVWF        intdex1
d34lop  decfsz      intdex1,1
        goto        d34lop
        nop
        return
dely83  MOVLW       0x1D            ;adjust to match baud rate
        nop
        nop
        goto        delex

;*******SUB_HEX*******

;get the hi-nibble
hex     nop
hexhi   call        rxchar          ;Read hi-nibble 0-F Hex
        MOVF        rxdata,0
        MOVWF       byte            ;copy new data into BYTE
        MOVWF       rxtemp          ;and temp file.
        MOVLW       0x30
        ANDWF       rxtemp,0
        xorlw       0x30
        btfsc       status,z        ;Wait for a number
        goto        shift

;If not a number now test for a letter
        MOVF        byte,0          ;Copy nibble back
        MOVWF       rxtemp        ;to temp file.
        MOVLW       0x40
        ANDWF              rxtemp,0
        Xorlw              0x40
        btfss       status,z      ;If this is a letter continue, else try again
        goto        hexhi
        MOVLW       0x01
        ADDWF       byte,1        ;Add 1 to ASCII
        BSF         byte,3        ;Now convert to hex nibble A-F

shift   SWAPF       byte,1
        MOVLW       0xF0          ;mask off lower nibble
        ANDWF       byte,1        ;BYTE is now upper nibble, range 0-Fh
        call        ok


;Get the lo_nibble

hexlo   call        rxchar        ;Read lo-nibble 0-F Hex
        MOVF        rxdata,0
        MOVWF       nibble        ;copy new data into NIBBLE
        MOVWF       rxtemp        ;and temp file.
        MOVLW       0x30
        ANDWF       rxtemp,0
        xorlw       0x30
        btfsc       status,z      ;If not a number look for a letter
        goto               join

;If not a number now test for a letter

        MOVF        nibble,0      ;Copy reprate byte
        MOVWF       rxtemp        ;back to temp file.
        MOVLW       0x40
        ANDWF       rxtemp,0
        xorlw       0x40
        btfss       status,z      ;If not letter or a number try again
        goto        hexlo
        MOVLW       0x01
        ADDWF       nibble,1      ;Add 1 to ASCII
        BSF         nibble,3      ;Now convert to hex nibble A-F

join    MOVLW       0x0F          ;mask off upper nibble
        ANDWF       nibble,0      ;NIBBLE is now lower nibble, range 0-Fh
        ADDWF       byte,1        ;BYTE becomes the full hex byte
        movlw              0x00
        movwf              rxdata        ;clear characters which can be read in
Main loop
        call        ok            ;ie C,D,E or F
        return
```

```
;*******SUB-PULSEH(Multiplier)*****

pulseh  call        hex
        movf        byte,0
        movwf       HNUM
        return

;*******SUB-PULSEL*****

pulsel  call        hex
        movf        byte,0
        movwf       LNUM
        return

;*******SUB-DELAYH*****

delayh  call        hex
        movf        byte,0
        movwf       HDEL
        return

;*******SUB-DELAYL*****

delayl  call        hex
        movf        byte,0
        movwf       LDEL
        return

;*******SUB-WIDTH*******

width   call        short
        rlf         rxnum,f
        movlw       b'00001110'
        andwf       rxnum,f
        movf        rxnum,w
        MOVWF       pwidth
        movwf       porta         ;output combination
        call        ok
        return

;*******SUB_SEQ****loop escape with 'X'*******

seq     call        ok
        movlw       b'10010000'   ;Turn on internal UART to receive
        movwf       rcsta

        movf        LEDnum,w      ;Select LED
        movwf       portb
        movf        HNUM,w        ;Transfer number of pulses high byte
        movwf       THNUM         ;to temp reg
```

```
nextn   movf        LNUM,w      ;Transfer number of pulses low byte
        movwf       TLNUM       ;to temp reg

        bsf         porta,5     ;Active LED on, Ready LED off

nextp   bsf         porte,2     ;enable LED and driver
        nop
        bsf         porte,1     ;Output pulse ,approx 2us.Note bit set used
        bcf         porte,2     ;Disable LED driver
        nop
        nop
        nop
        nop                     ;wait for current to drain
        nop
        nop
        nop
        bcf         porte,1     ;now safe to end pulse sequence
        movf        HDEL,w      ;Transfer delay loop high byte
        movwf       THDEL       ;to temp reg
nextd   movf        LDEL,w      ;Transfer delay loop low byte
        movwf       TLDEL       ;to temp reg

inhib   btfss       porte,0     ;test hardware inhibit (low=inhibit)
        call        inhibit     ;loop until inhibit=1 or UFIN

        movf        rcreg,w     ;Stop all addressed pulsers?
        xorlw       0x5F        ;compare with '_'
        btfsc       status,z
        goto        Xfin

;       call    delayb           ;Adjust delay to set period (not in use)

        decfsz      TLDEL,f
        goto        inhib       ;loop until delay low byte is zero
        decfsz      THDEL,f
        goto        nextd       ;loop until delay high byte is zero

        decfsz      TLNUM,f
        goto        nextp       ;loop until number low byte is zero
        decfsz      THNUM,f
        goto        nextn       ;loop until number highbyte is zero
        goto        fin         ;End of sequence

Xfin    movf        rcreg,w     ;wait for 'X'
        xorlw       0x58
        btfss       status,z
        goto        Xfin

fin     bcf         porte,2     ;Disable LED driver
        movlw       0x00        ;turn off UART
        movwf       rcsta
        call        ok
        return

;*******SUB_EXTRUN****loop escape with '_X'*******

extrun  call        ok
        movlw       b'10010000'   ;Turn on internal UART to receive
        movwf       rcsta

        movf        LEDnum,w      ;Select LED
        movwf       portb
        movf        HNUM,w        ;Transfer number of pulses high byte
        movwf       THNUM         ;to temp reg

        bcf         status,rp1
        bsf         status,rp0
        movlw       0x00
        movwf       trise         ;Temporary enable Inhibit line as output
        bcf         status,rp0
        bcf         status,rp1
        bcf         porte,0       ;Inhibit pulsing
        bsf         porta,5       ;Active LED on, Ready LED off
        bsf         portd,5       ;enable external trigger
        nop
        nop
        nop
        bsf         porte,0       ;Enable pulsing
        bcf         status,rp1
        bsf         status,rp0
        movlw       0x01
        movwf       trise         ;Re-enable Inhibit line as input
        bcf         status,rp0
        bcf         status,rp1

reload  movf        LNUM,w        ;Transfer number of pulses low byte
        movwf       TLNUM         ;to temp reg

extrig  btfsc       portd,4       ;look for trigger input low
        goto        chk
        btfss       portd,4
        goto        nxtopt
chk     btfss       porte,0       ;test hardware inhibit (low=inhibit)
        call        inhibit       ;loop until inhibit=1 or UFIN

        movf        rcreg,w       ;Stop all addressed pulsers?
        xorlw       0x5F          ;if '_' is received.
        btfsc       status,z
        goto        finX
        goto        extrig
```

```
nxtopt  call        delayb
        decfsz      TLNUM,f
        goto        extrig       ;go back to wait for next trigger pulse
        decfsz      THNUM,f
        goto        reload       ;not end of sequence yet!
        goto        finS         ;Trigger count limit, so exit routine.

finX    movf        rcreg,w      ;wait for 'X'
        xorlw       0x58
        btfss       status,z
        goto        finX

finS    bcf         porta,5      ;Active LED off
        bcf         portd,5      ;disable external trigger
        movlw       0x00         ;turn off UART
        movwf       rcsta
        call        ok            ;send 'K' for end of sequence
        return

;*******CONTRIG**********

ctrig   call        ok
        movlw       b'10010000'  ;Turn on internal UART to receive
        movwf       rcsta

        movf        LEDnum,w     ;Select LED
        movwf       portb
        bsf         porta,5      ;Active LED on, Ready LED off
        bsf         portd,5      ;enable external trigger
        bsf         porte,0      ;Enable pulsing

tstex   btfss       porte,0      ;test hardware inhibit (low=inhibit)
        call        inhibit      ;loop until inhibit=1 or UFIN

        movf        rcreg,w      ;Stop all addressed pulsers?
        xorlw       0x5F         ;if '_' is received.
        btfsc       status,z
        goto        fini
        goto        tstex

fini    movf        rcreg,w      ;wait for 'X'
        xorlw       0x58
        btfss       status,z
        goto        fini
        bcf         porta,5      ;Active LED off
        bcf         portd,5      ;disable external trigger
        movlw       0x00         ;turn off UART
        movwf       rcsta
        call        ok           ;send 'K' for end of sequence
        return

;*******DELAY************

delayb  MOVLW       0xFF            ;Minimum delay for period if reqd.
        MOVWF       sdely
dly     DECFSZ      sdely,F
        GOTO        dly
        return

;*******SUB_CONTIN****loop escape with <nul X>*******

contin  call        ok
        movlw       b'10010000'     ;Turn on internal UART to receive
        movwf       rcsta
        movf        LEDnum,w
        movwf       portb           ;Select active LED
infin   movf        HNUM,w          ;Transfer number of pulses high byte
        movwf       THNUM           ;to temp reg
nxtn    movf        LNUM,w          ;Transfer number of pulses low byte
        movwf       TLNUM           ;to temp reg

        bsf         porta,5         ;Active LED on, Ready LED off

nxtp    bsf         porte,2         ;enable LED driver
        nop
        bsf         porte,1         ;Output pulse ,approx 2us.Note bit set used
        bcf         porte,2         ;Disable LED driver
        nop
        nop
        nop
        nop                         ;wait for current to drain
        nop
        nop
        nop
        bcf         porte,1         ;now safe to end pulse sequence
        movf        HDEL,w          ;Transfer delay loop high byte
        movwf       THDEL           ;to temp reg
nxtd    movf        LDEL,w          ;Transfer delay loop low byte
        movwf       TLDEL           ;to temp reg

ihib    btfss       porte,0         ;test hardware inhibit (low=inhibit)
        call        inhibit         ;loop until inhibit=1 or UFIN

        movf        rcreg,w         ;Stop all addressed pulsers?
        xorlw       0x5F            ;compare with '_'
        btfsc       status,z
        goto        Yfin

;       call        delayb          ;Adjust delay to set period (not used)
```

```
        decfsz    TLDEL,f
        goto      ihib        ;loop until delay low byte is zero
        decfsz    THDEL,f
        goto      nxtd        ;loop until delay high byte is zero

        decfsz    TLNUM,f
        goto      nxtp        ;loop until number low byte is zero
        decfsz    THNUM,f
        goto      nxtn        ;loop until number highbyte is zero
        goto      infin       ;End of sequence

Yfin    movf      rcreg,w     ;wait for 'X'
        xorlw     0x58
        btfss     status,z
        goto      Yfin
        movlw     0x00        ;turn off UART
        movwf     rcsta
        call      ok
        return

;*******SUB-INHIBIT******

inhibit bcf       porta,5     ;Pulsing indicator LED off
        movlw     0x27        ;turn off 'Active LED'
        movwf     portb
test0   btfss     porte,0
        goto      test0
        movf      LEDnum,w
        movwf     portb
        bsf       porta,5     ;Pulsing indicator LED on
        return

;*******SUB-TEMPH********

temph   movf      adresh,w    ;msb of temperature - in bank 0
        movwf     txdata
        call      txchar      ;send msb
        return

;*******SUB-TEMPL*******

templ   bsf       status,rp0  ;select bank 1 to read out lsb
        bcf       status,rp1
        movf      adresl,w    ;lsb of ad reading - in bank 1
        bcf       status,rp0  ;select bank 0
        bcf       status,rp1

        movwf     txdata
        call      txchar      ;send lsb
        return
```

```
;*******SUB-READ A/D*****

readad bsf          status,rp0      ;select bank 1
       bcf          status,rp1
       movlw        b'10001110'     ;RA0 set as a/d input
       movwf        adcon1

       bcf          status,rp0      ;select bank zero
       bcf          status,rp1
       movlw        b'10000001'     ;Fosc32, select ad0 (RA0)
       movwf        adcon0
       call         msdelay         ;1mS sample delay
       bsf          adcon0,2        ;start conversion
       call         msdelay         ;1mS delay, conversion complete
                                    ;when bit 2 of adcon0 is clear
chkconbtfsc         adcon0,2        ;bit 2 clear=conversion done
       goto         chkcon
       call         ok
       return                       ;ad value in adresh and adresl
                                    ;000000vv   vvvvvvvv

;*******SUB-1mS DELAY******

msdelay movlw       d'166'          ;1mS delay
       movwf d
msloop nop
       nop
       nop
       decfsz  d,1
       goto         msloop
       return

;*******SUB_LOAD DAC*****

ldac   movf         curnth,w
       movwf        chtmp
       movf         curntl,w
       movwf        cltmp
       bsf          PORTD,0
       bsf          PORTD,2
       bsf          PORTD,3
       MOVLW        0x02            ;setup loop for 2 characters
       MOVWF        intdex
       bcf          STATUS,C
       rlf          chtmp,f         ;shift out msb for tx
oplop1 bcf          PORTD,0         ;set clock low
       btfss        STATUS,C
       bcf          PORTD,1         ;set data low
       btfsc        STATUS,C
```

```
        bsf     PORTD,1     ;set data high
        bsf     PORTD,0     ;clock data into shift register
        rlf     chtmp,f     ;next data bit
        decfsz  intdex,1
        goto    oplop1
        MOVLW   0x08        ;setup loop for 8 characters
        MOVWF   intdex
        bcf     STATUS,C
        rlf     cltmp,f     ;shift out msb for tx
oplop2  bcf     PORTD,0     ;set clock low
        btfss   STATUS,C
        bcf     PORTD,1     ;set data low
        btfsc   STATUS,C
        bsf     PORTD,1     ;set data high
        bsf     PORTD,0     ;clock data into shift register
        rlf     cltmp,f     ;next data bit
        decfsz  intdex,1
        goto    oplop2
        bcf     PORTD,2     ;load data
        bsf     PORTD,2
        call    ok
        return

;--------------------------------------------------------------

Start   call    nxchar      ;Clear firmware UART

Main    bcf     PORTA,5     ;Ready LED on, Active LED off

        MOVF    boxnum,w
        MOVWF   tmpbox
        call    rxchar
        MOVF    rxdata,w
        xorwf   tmpbox,w    ;compare to see if box is addressed
        BTFSS   status,z
        goto    Main        ;continue looping until addressed
        nop
        call    rxchar      ;check new character
        MOVF    rxdata,w
        xorlw   0x41
        btfsc   status,z    ;jump if not 'A'
        call    ok
        MOVF    rxdata,0
        xorlw   0x42
        btfsc   status,z    ;jump if not'B'
        call    heighth
        MOVF    rxdata,0
        xorlw   0x43
        btfsc   status,z    ;jump if not'C'
        call    heightl
```

```
MOVF        rxdata,0
xorlw       0x44
btfsc       status,z        ;jump if not'D'
call        width
MOVF        rxdata,0
xorlw       0x45
btfsc       status,z        ;jump if not'E'
call        select
MOVF        rxdata,0
xorlw       0x46
btfsc       status,z        ;jump if not'F'
call        pulseh
MOVF        rxdata,0
xorlw       0x47
btfsc       status,z        ;jump if not'G'
call        pulsel
MOVF        rxdata,0
xorlw       0x48
btfsc       status,z        ;jump if not'H'
call        delayh
MOVF        rxdata,0
xorlw       0x49
btfsc       status,z        ;jump if not'I'
call        delayl
MOVF        rxdata,0
xorlw       0x4A
btfsc       status,z        ;jump if not'J'
call        seq
MOVF        rxdata,0
xorlw       0x4B
btfsc       status,z        ;jump if not'K'
call        contin
MOVF        rxdata,0
xorlw       0x4C
btfsc       status,z        ;jump if not'L'
call        readad
MOVF        rxdata,0
xorlw       0x4D
btfsc       status,z        ;jump if not'M'
call        temph
MOVF        rxdata,0
xorlw       0x4E
btfsc       status,z        ;jump if not'N'
call        templ
MOVF        rxdata,0
xorlw       0x4F
btfsc       status,z        ;jump if not'O'
call        ldac
MOVF        rxdata,0
xorlw       0x50
```

```
        btfsc       status,z        ;jump if not'P'
        call        extrun
        MOVF        rxdata,0
        xorlw       0x51
        btfsc       status,z        ;jump if not'Q'
        call        ctrig
        goto        Main            ;wait to be addressed

;------------------------------------------------------------
        END
```